## REMARKS

Reconsideration and allowance are respectfully requested.

Claims 43-62 stand rejected under 35 U.S.C. §101 as allegedly being directed to non-statutory subject matter. This rejection is respectfully traversed.

Claim 43 now recites: "A computer program product including instruction code embodied in a medium readable by a data processing apparatus, which when executed by the data processing apparatus, controls the data processing apparatus to execute a sequence of variable length instructions stored within a plurality of discrete memory address regions within a memory of said data processing apparatus." The claim is not to a program per se. The product includes instruction code embodied in a medium readable by a data processing apparatus. There is no requirement under 35 U.S.C. §101 that "specific elements of the program product be reflected into the claim body." Moreover, specific code elements are in fact recited in the body of claim 43.

Applicants traverse the Examiner's conclusion that "no useful, concrete and tangible results can be determined." Claims 43-62 are concerned with how a data processing apparatus handles fragmented instructions (page 2, line 4-7 and page 3, lines 1-3). Although fragmented instructions can be desirable (e.g. for enhanced security), they also are associated problems such as memory aborts (see page 8, lines 21-24). The computer program product recited in claims 43-62 enables a data processing apparatus to handle such situations and thus to operate more efficiently. Certainly, given the central role that data processing apparatus have in daily activities of most people in the industrialized world, enabling data processing apparatus to operate more efficiently is a useful, concrete, and tangible result of claims 43-62.

Most of the claims stand rejected for anticipation under 35 U.S.C. §102 based on USP 5,598,544 to Ohshima. This rejection is respectfully traversed.

To establish that a claim is anticipated, the Examiner must point out where each and every limitation in the claim is found in a single prior art reference. *Scripps Clinic & Research Found. v. Genentec, Inc.*, 927 F.2d 1565 (Fed. Cir. 1991). Every limitation contained in the claims must be present in the reference, and if even one limitation is missing from the reference, then it does not anticipate the claim. *Kloster Speedsteel AB v. Crucible, Inc.*, 793 F.2d 1565 (Fed. Cir. 1986). Ohshima fails to satisfy this rigorous standard.

The technology of this application is directed to data processing systems that can execute a variable length instruction stored in separate memory locations. The first part of a variable length instruction is stored in a first memory location, and the second part of the instruction is stored in a second separate and different memory location. A "fix-up" memory address region is used to bring together the separate parts of the single instruction. When a separated variable length instruction occurs program execution flow reading from "normal" memory, the program execution flow is temporarily diverted to the fix-up memory to read the freshly reconstructed variable length instruction stored there. Thereafter, the program execution flow is returned to reading from the normal memory.

Ohshima is concerned with a very different problem: efficient processing of an instruction that includes both basic and expanded segments. A basic segment contains a code indicating the type of instruction (basic or expanded), and an expanded segment contains information relevant to the type of instruction specified by the basic segment. One instruction is formed from one or more basic and expanded segments. See column 3, lines 31-40. Even though the basic segments have a known length, the length of expanded segments can vary, and

this length is not known until its corresponding basic segment has been decoded. Hence, in a situation where one of Ohshima's instructions consists of two basic segments and the first basic segment is followed by an expanded segment, the second basic segment cannot be input into a decoder until after the first basic segment has been decoded, which allows the length of the expanded segment to be determined. Thus, two cycles are required to decode the single instruction. See column 2, line 59 to column 3, line 9.

Ohshima's instruction buffer device includes a pre-decoder 15 in Figure 6A, which identifies the segments within a single instruction code string and stores control markings in a parallel storage section 16. The markings indicate the "nature" of each segment. See column 7, lines 54 to column 8, line 9 and enables the instruction buffer device to rearrange the segments 8 as they are transferred to the instruction code bus. So the two basic segment instruction in the situation described above occupies the first and second positions, and any expanded segments occupy subsequent positions. Figures 6B and 7B show the results of the rearrangement where basic segment 1 field X1 is next to basic segment 2 field X2 followed by expanded segment fields Y1 and Y2. Hence the decoding of the two basic segments 9 in Figure 6B may then occur in one cycle.

Given that Ohshima does not even consider the problem of executing variable length instructions which span two discrete memory address regions, it is no surprise that Ohshima fails to disclose a number of features recited in the independent claims. First, the Examiner fails to show where Ohshima describes "a sequence of variable length instructions stored within a plurality of discrete memory address regions within a memory," as recited in claim 1. All that Ohshima discloses is an instruction code being fetched from "external memory or cache" to a "code storage section" which is 4 blocks wide 403 in Figure 4 or 8 blocks wide 3 in Figure 6A.

Where are the discrete and separate memory locations storing different parts of a variable length instruction in Ohshima?

Second, if the Examiner is contending that the rows of the code storage section (4, 4', 4" etc. in Figures 6A and 7A) constitute "discrete memory address regions," then Ohshima fails to disclose "detecting an attempt to execute a variable length instruction spanning two discrete memory address regions," recited in step (i) of claim 1. Ohshima does not treat the rows of the code storage section as discretely addressable entities in memory, but rather joins them in a continuous loop as indicated by the arrows in Figure 7A and as described in column 8, lines 16-23. In addition, Ohshima rearranges segments within the single instruction regardless of whether a particular instruction happens to overrun from one row of the code storage section into the next.

Third, Ohshima lacks step (ii) of claim 1. Here, the Examiner refers to the readout pointer described in column 2, lines 38-50 in conjunction with the wrap-around feature outlined in the previous paragraph. Reading-out of the code storage section puts the rearranged segments of the instruction onto the instruction code bus. See column 8, lines 63-65 and Figure 6B. Consequently, Ohshima fails to disclose "concatenating instruction data...into a fix-up memory address region of said memory." Additionally, Applicants disagree with the Examiner's contention that Figures 1, 5, and 7B illustrate concatenation of instruction fields. These figures merely illustrate the order of the constituent segments within a single instruction field being rearranged.

Fourth, steps (iii) and (iv) recite "diverting program execution flow to execute said current variable length instruction from within said concatenated instruction data in said fix-up memory address region" and "restoring program execution flow to execute instructions

following said variable length instruction from within said following memory address region," respectively. The Examiner fails to detail how Ohshima diverts or restores execution flow but merely refers to the fact that Ohshima has an execution unit 13. Of course, all instructions in Ohshima (whether in prior art Figure 4 or Ohshima's invention shown in Figure 6B) are executed from the instruction code bus, regardless from where they originate. Ohshima simply rearranges the order of the segments of an instruction before they are put onto that instruction code bus. There is no "fix-up memory address region of the memory." For the same reasons, Ohshima fails to disclose "restoring program execution flow to execute instructions following said variable length instruction from within said following memory address region." No such execution flow restoration is needed.

The application is in condition for allowance. An early notice to that effect is requested.

Respectfully submitted,

**NIXON & VANDERHYE P.C.**

By: ______________________
John R. Lastova
Reg. No. 33,149

JRL:sd
901 North Glebe Road, 11th Floor
Arlington, VA 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100